Celadon Trusty Hardware Binding Developer Guide

Intel OTC Android Security Team
Version 1.0

Table of Contents

Revision History

Date	Version	Description
Oct 24, 2018	0.3	Initial Draft
Nov 28, 2018	0.5	Internal review complete
Dec 02, 2018	0.51	Added the platform dependencies
Dec 19, 2018	1.0	Finalize the review

1. Introduction

1.1. Purpose and Scope of the document

This document covers the guide for general hardware binding for Trusty (TEE) on Intel platforms powered by Intel Celadon open source project for Android. It contains the requirement overview, RoT (Root of Trust) sources for hardware binding, security guideline and an integration guide for TPM based hardware binding which is implemented as a developer reference in Celadon.

1.2. Target Audience

The target audience includes design engineers and developers (especially for BIOS and Boot loader developers). This document should enable the design engineers and developers to come up with the effort estimates, design and implementation of the hardware binding feature for Android product based on Celadon.

1.3. Terminology

Acronym	Description
AOSP	Android Open Source Project
AVB	Android Verified Boot
BIOS	Firmware for Basic Input/Output System.
CDD	Compatibility Definition Document for Android
CSE	Converged Security Engine
FW	Firmware
EOM	End of Manufacturing
HuK	Hardware Unique Key
NV	Non-Volatile
NVMe	Non-Volatile Memory express
OTP	One Time Programming (fuses)
PTT	Intel Platform Trust Technology
RoT	Root of Trust

RPMB	Replay Protected Memory Block
SPE	Secure Platform Engine
TCB	Trusted Computing Base
TEE	Trusted Execution Environment (Trusty)
TPM	Trusted Platform Module. In this document, TPM 2.0 is used until an explicit version is specified.
UFS	Universal Flash Storage (UFS) is a flash storage specification for digital cameras, mobile phones and consumer electronic devices.
UEFI	Unified Extensible Firmware Interface

2. Hardware Binding Overview

2.1. Android Security Requirements

Android Compatibility Definition Document (CDD) defines the security requirements for “Keys and Credentials” which requires a hardware-backed isolated execution environment for keystore implementation.

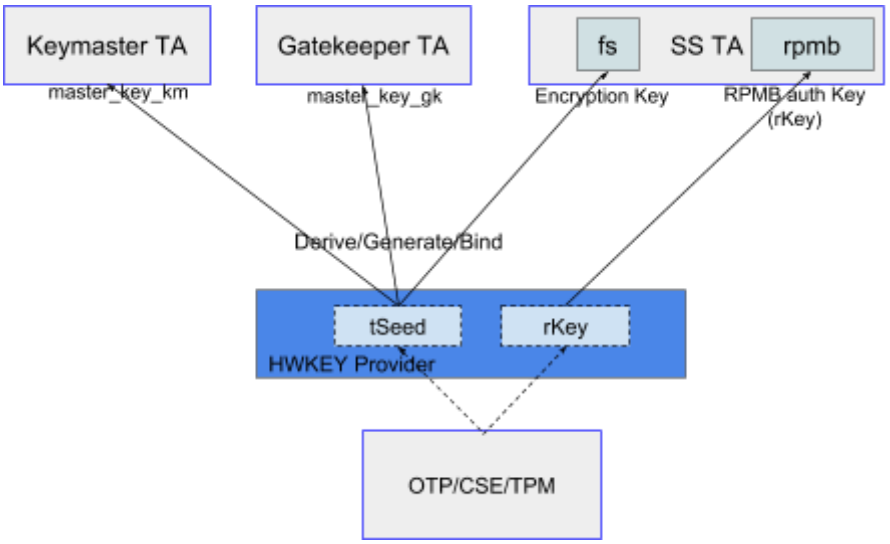
When the device implementation supports a secure lock screen, it:

- *[C-1-1] MUST back up the **keystore implementation** with an **isolated execution environment**.*
- *[C-1-2] MUST have implementations of RSA, AES, ECDSA and HMAC cryptographic algorithms and MD5, SHA1, and SHA-2 family hash functions to properly support the Android Keystore system's supported algorithms in an area that is **securely isolated** from the code running on the kernel and above. Secure isolation MUST block all potential mechanisms by which kernel or user space code might access the internal state of the isolated environment, including DMA. The upstream Android Open Source Project (AOSP) meets this requirement by using the Trusty implementation, but another ARM TrustZone-based solution or a third-party reviewed secure implementation of a proper hypervisor-based isolation are alternative options.*
- *[C-1-3] MUST perform the lock screen authentication in the **isolated execution environment** and only when successful, allow the authentication-bound keys to be used. Lock screen credentials MUST be stored in a way that allows only the isolated execution environment to perform lock screen authentication. The upstream Android Open Source Project provides the Gatekeeper Hardware Abstraction Layer (HAL) and Trusty, which can be used to satisfy this requirement.*

The keys and credentials implementation with an isolated execution environment requires a HuK (Hardware Unique Key) or a hardware-bound secret that is only accessible to TEE. The process of binding key derivations with this HuK/secret is called hardware binding.

2.2. Trusty Seed

The Secret derived from HuK that is required by Trusty (TEE), is called ‘Trusty Seed’ (tSeed). Trusty/TEE relies on platform to provide a high entropy seed that is used to derive / generate other various internal keys for Android data confidentiality / integrity protections.



2.3. RPMB Seed and authentication Key

RPMB (Replay Protected Memory Block) is one of partitions in eMMC/UFS/NVMe flash storage. Some storage device like legacy SATA hard drive doesn't have RPMB.

RPMB is used to implement the Trusty Secure Storage for data confidentiality, integrity, and replay protection. In order to use RPMB, a RPMB authentication key (rKey) is required and to be programmed before the first use. In TPM-based RoT implementation, rKey is also stored in TPM, and well protected with appropriate policy of access control. In this document, RPMB Seed (rSeed) and RPMB Authentication Key (rKey) may be used interchangeably. But in a specific implementation, the rKey could be either equal to rSeed, or derived from rSeed (e.g. rKey = KDF (rSeed, Device ID#)).

There are two cases out of the scope of this document.

- For storage device in a platform that doesn't have RPMB, a software simulated RPMB with a dedicated ordinary logical user data partition is provided in Celadon source repository.
  In this case, it cannot provide replay and integrity protection.

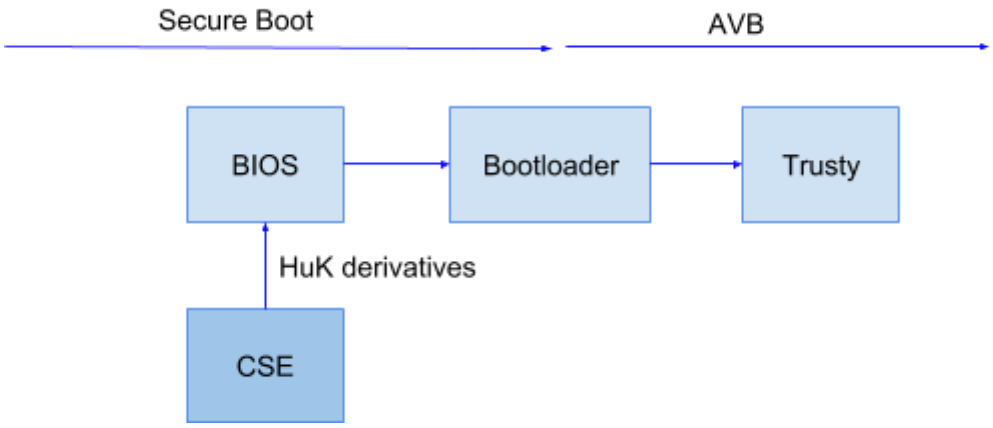
- In platforms where Trusty cannot own the RPMB partition or the authentication rKey, for instance, if CSE firmware is booting from UFS storage device, it will take the ownership of a RPMB partition as its own secure storage media. In this particular case, Trusty is not able to get the rKey from CSE FW to access the RPMB partition. In order to solve this issue, it requires either a RPMB sharing with CSE FW (CSE FW and BIOS FW modification are required) or a flash storage drive with two and more RPMB partitions, like a UFS3.0 drive or NVMe drive.

3. Root of Trust Sources

The Root of Trust (RoT) must reside in hardware and be protected by hardware. Typical RoT sources in Intel platform include CSE(Converged Security Engine), OTP (one time programming) fuses, and TPM (Trusted Platform Module). Different implementation and different platform configuration determine different solution that is used for hardware binding. But no matter what implementation are chose as RoT, secure boot and AVB (Android Verified Boot) must be enabled to make sure the boot chain is trusted.

3.1. CSE Firmware

CSE is a coprocessor security engine running on many Intel platforms. On Intel Atom-based platform, CSE FW provides a platform unique Trusty Seed as well as a secure way to retrieve the seed by BIOS. BIOS thereafter hands over the seed to bootloader and then the bootloader passes it to Trusty. For UEFI based on BIOS, it is recommended to cache the Trusty seed and provide a customized UEFI protocol for bootloader to retrieve this Seed during boot time when UEFI Secure Boot is enabled. An example flow would be like this below.



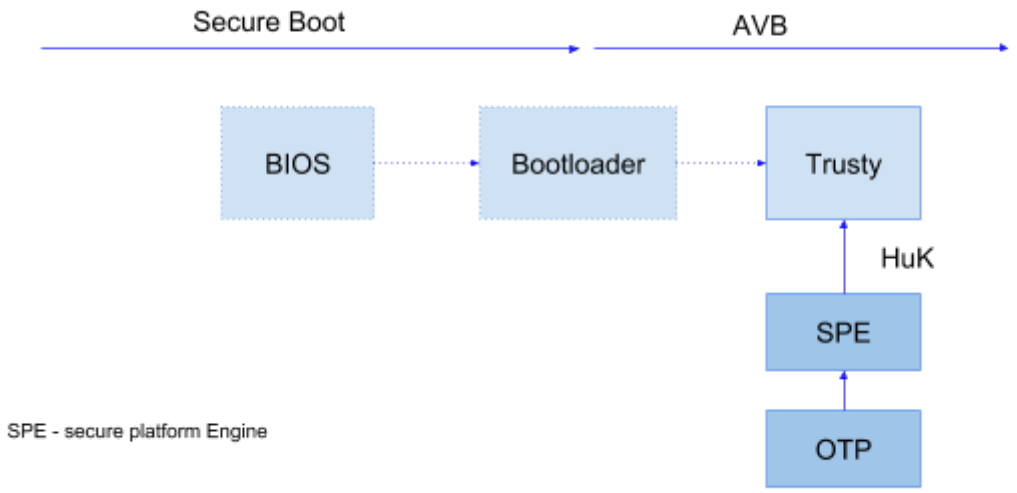
CSE rooted RoT handover flow

The Trusty seed originated from CSE depends on support from CSE FW firmware and BIOS. As of the time we write this document, the CSE FW on Kaby Lake NUC platform doesn't support Trusty seed yet. So the other solution can be used for this purpose, e.g. TPM-based Seed, which will be described as below.

3.2. OTP Fuses

In a platform that has a dedicated secure engine for Trusty, then a dedicated OTP fuse can be allocated to store the Trusty seed. And the platform security configuration must ensure that only Trusty/TEE is allowed to talk with the secure engine.

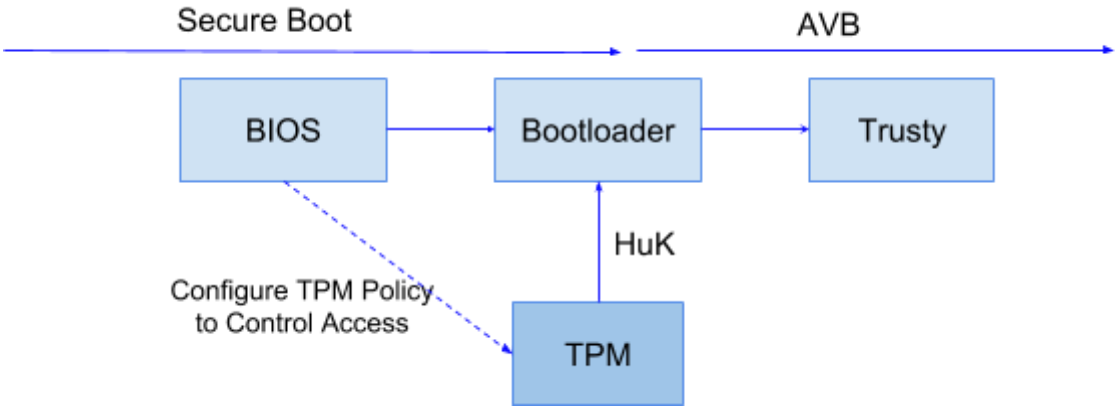
In this case, the access to OTP fuses are restricted to SPE (secure platform engine) only and only Trusty can retrieve the seed through the SPE. An implementation boot flow would be like this diagram below.



OTP fuses rooted RoT handover flow

3.3. TPM secure NV index

TPM provides the non-volatile storage (NV index) with policy for access control. The NV index can be leveraged to host the secrets and a proper policy makes sure the secrets cannot be accessed by untrusted entities. TPM 2.0 provides three hierarchies for NV indexes, Owner Hierarchy and Platform Hierarchy will serve for different cases in this document.



TPM rooted RoT handover flow

The implementation of hardware bindings on TPM is implemented in Celadon. In this document, we focus on the hardware binding implementation based on TPM.

4. TPM Rooted Seed Solution Dependencies

The TPM rooted seed solution, including Trusty Seed and RPMB Seed/Key, depends on below hardware and firmware/BIOS support:

a. TPM 2.0 support (TPM 1.2 not supported), i.e., Platform Trust Technology (PTT a.k.a. fTPM) functionality, -or-, a discrete TPM 2.0
b. IFWI supports TCG measured boot profiles (owner hierarchy only)
c. BIOS supports UEFI profile of TCG specs for PC

On the current Celadon supported platforms, Kaby Lake NUC and Apollo Lake NUC, the devices all provide the TPM 2.0 support:

- KBL-NUC [NUC7i5BNH-IDD]
- APL-NUC [NUC6CAYH]

5. TPM Rooted Trusty Seed

This subsection has much details of the TPM-based implementation because TPM2.0 is generally available in all the Celadon reference platforms. E.g. CSE FW provides PTT which is TCG TPM2.0 compliant implementation in Intel platforms.

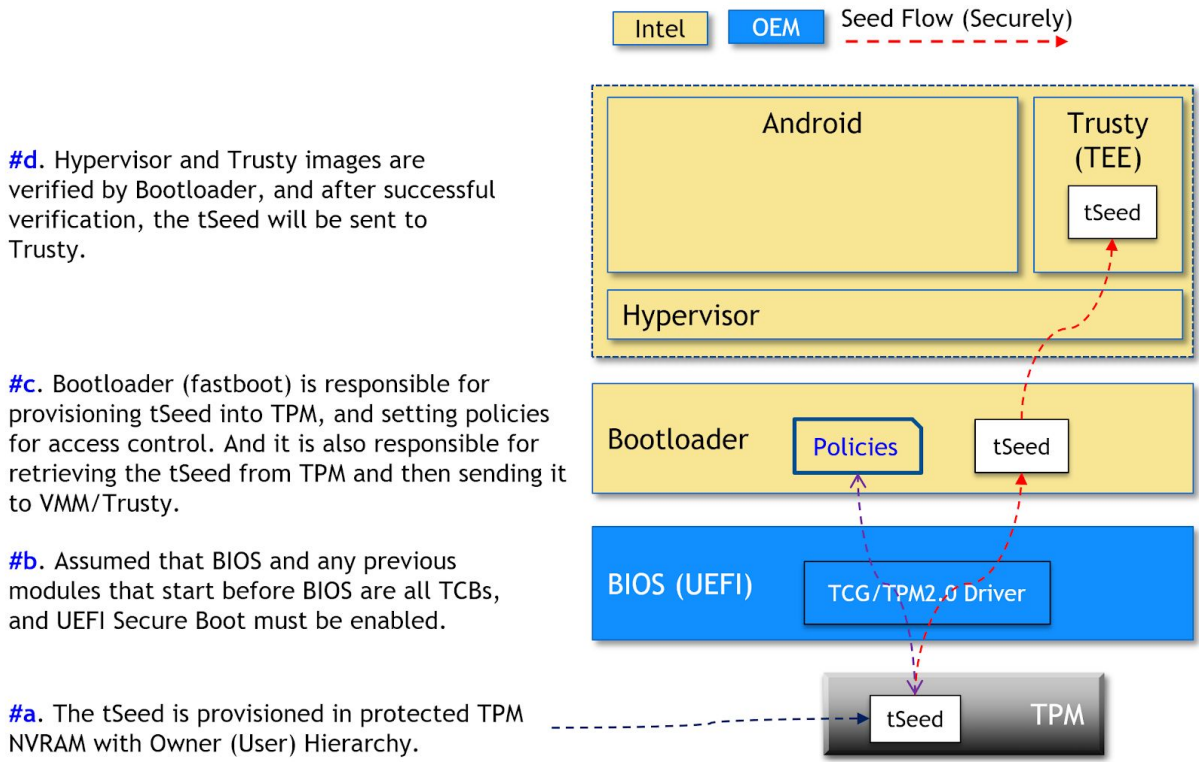
TPM provides the non-volatile storage (NV index, or NVRAM) capability along with capability of applying policy for access control. In order to meet the requirements of Trusty Seed, the Trusty/TEE Seed shall be 256-bit long at least, be unique/random per platform, and access to this Seed is restricted by TPM policy and UEFI secure boot.

In TPM 2.0, a NV Index can be created with either Platform Hierarchy or Owner Hierarchy. In order to understand the differences of between them, please dive into more details in TPM2.0 specification documents.

In this document (as well as the reference implementation in Bootloader), TPM Owner Hierarchy is used to create NV index storing the Trusty Seed secret. This is primarily because TPM Platform Hierarchy is disabled or locked by UEFI BIOS. It means that only BIOS is allowed to use TPM Platform Hierarchy.

5.1. TPM-Rooted Trusty Seed with Owner Hierarchy

The TPM-rooted Trusty/TEE Seed (tSeed) handoff flow is illustrated as the diagram below.



Trusty Seed (tSeed) secure handoff with Owner Hierarchy

In this workflow, a tSeed is provisioned initially with a random 256-bit value into a TPM NV index with TPM Owner Hierarchy, then Bootloader sets appropriate access control policy (see later) to restrict any illegitimate access to this secret. In every boot, Bootloader retrieves this secret when access policy is satisfied, and sends it to Trusty after successful Trusty/VMM image verification.

TPM NV Index Attributes

When creating a new NV Index with TPM2 command TPM2_NV_DefineSpace(), the attributes of this NV Index for Trusty Seed shall be be configured as indicated in following table.

Note those attributes can only be configured during the creation of the NV index, in other words, as long as they are created, there is no interface for software to change/update the attributes of TPM NV index.

Attribute Bit(s)	Value	Descriptions
TPMA_NV_POLICYREAD	1	The Index data may be read if the authPolicy is satisfied.
TPMA_NV_POLICYWRITE	1	Authorizations to change the Index contents that require USER role may be provided with a policy session.
TPMA_NV_WRITEALL	1	A partial write of the Index data is not allowed. The

		write size shall match the defined space size.
TPMA_NV_WRITEDEFINE	1	TPM2_NV_WriteLock() may be used to prevent further writes to this location.
TPMA_NV_READ_STCLEAR	1	TPM2_NV_ReadLock() may be used to SET TPMA_NV_READLOCKED for this Index.

Note all other attribute bit field values should be clear/0.

NV Index Access Control Policy

Access	Policy Description
Read	After each TPM power cycle, read is allowed once and then the NV index is locked (until next power cycle). Also the access to tSeed is associated with TPM2.0 PCR Policy with PCR[7] value. **Note** according to TCG TPM2 PC-compliant specification and UEFI Secure Boot Specification, the PCR[7] measures the Secure Boot policies / configurations (e.g. Secure Boot Variables, PK/KEK/DB/DBx) by BIOS.
Write	Write once during tSeed provisioning, and then any write access is not allowed permanently (unless NV Index is deleted), regardless of TPM power cycles.
Deletion	Policy above only controls read and write access. TPM NV index created with TPM Owner Hierarchy can be deleted (or 'undefined' in TPM's term) in these two approaches regardless of read/write access control policies: a. TPM2_Clear() Only BIOS is allowed to invoke this command, but users can request BIOS to do this with user's physical presence. Hence BIOS can be customised to disable this interface for physical presence request. Or, BIOS can completely disable the invocation of this TPM command with TPM2_ClearControl(). **Note** this may be optional because some project might require TPM2_Clear() to do factory reset to clear all user data protected by Android/Trusty, in this case then a new tSeed should be re-generated / re-provisioned / and protected. b. TPM2_NV_UndefineSpace() By default, the TPM Owner / User AuthValue is zero (Well-Known

	Secret), so any software can delete / undefined any TPM NV Indices created with Owner Hierarchy with this default well-known AuthValue regardless of read/write access control policies. So the policy here is - Bootloader uses TPM2_HierarchyChangeAuth() command to change the Well-Known AuthValue (zeros) to a random high-entropy number, and then throw it away. This can permanently disable/lock the Owner Hierarchy because no one knows its new AuthValue (unless Owner Hierarchy is reset by TPM2_Clear() command).

5.2. Security Guidelines

This section is intended to depict the security guidelines and considerations including security assets, security dependencies, the threats and mitigations.

Security Assets

The asset here is Trusty Seed (tSeed) stored in TPM NV Index with Owner Hierarchy, and all the derivative keys from this tSeed.

The basic requirements for this rSeed asset are:

- It must be only accessible to the secure trusted entities. It should never be exposed to non secure world (e.g. Android OS including kernel).
- The tSeed temporarily buffered in the middle of the handoff shall be erased from memory after use as soon as possible, and it must not be stored in any ordinary storage on flash drive or disk.

Security Assumptions/Dependencies

It is assumed that all code/modules running before Bootloader are trusted and verified. Also, in Celadon project, since BIOS firmware is not fully controlled by Intel and it is developed by 3rd party OEM vendors for KBL/APL NUC devices on the market, we typically may do nothing but only provide our security mitigation recommendations to request their integration.

Threats and Mitigations

The table below lists all the identified threats and the corresponding mitigations:

#	Threat Descriptions	Mitigation(s)
1	BIOS image substitution.	BIOS firmware is TCB in this threat model.

		And, CSE FW Boot Guard enforces BIOS firmware image verification. (*This is out of our scope*)
2	Bootloader substitution with malicious image.	UEFI BIOS Secure Boot must be enabled. BIOS will refuse to start Bootloader with invalid signature.
3	Hypervisor / Trusty image substitution.	Bootloader refuses to boot any Hypervisor / Trusty image with invalid signature. And hence the tSeed won't be sent to malicious image.
4	User (again with physical presence) disables / changes the UEFI Secure Boot in BIOS configuration / setup screen.	tSeed read access policy is gated by PCR[7]. If UEFI Secure Boot is disabled / changed, the PCR[7] measurement will be different, then access to Seed will be denied by TPM.
5	User (again with Physical presence) configures BIOS to boot into alternative path like "UEFI shell" or any other UEFI applications, which then may allow to boot any other Osloader or OS.	1. OEM BIOS should always disable boot into "UEFI Shell" when UEFI Secure Boot is enabled / activated. This is a typical configuration for many platforms. 2. UEFI Shell (or other UEFI applications) also must be signed by OEM with trusted valid signature. And the PCR[7] should be extended at the beginning of this trusted UEFI Shell (or Applications) to prevent any further access to the Seed in TPM (even if the malicious Osloaders can be loaded).
6	Software with TPM Platform Hierarchy can have full control of TPM secrets.	According to TCG TPM2-compliant BIOS specification, Platform Hierarchy is either locked or disabled by BIOS before transferring control to next module in boot flow. So malicious software cannot have TPM Platform Hierarchy authorization.
7	Untrusted software (in Android) attempts to overwrite (or write) the tSeed NV index in TPM.	The TPM NV index policy forbids any write access after provisioning. (through TPM2_WriteLock() API)
8	Untrusted software (in Android) attempts to read the secret of tSeed NV index in TPM.	Read access is locked by Bootloader before transferring control to OS. Read policy is also gated by PCR[7] value (secure boot configurations)
9	DoS attack to remove / delete the tSeed	1. Disable TPM2_Clear() by BIOS

	NV Index from TPM (only viable when user has physical presence to the attacked device)	2. Disable or lock TPM Owner Hierarchy as mentioned previously. **Note** This might be an expected behaviors for some particular project / product that allows users to do factory reset.

5.3. Integration Guide

This section will guide the users how to integrate the specific hardware binding solution with Celadon reference implementation based on TPM2.0 capabilities.

In order to completely integrate this feature (TPM-rooted Trusty Seed implementation), the modifications are required in BIOS firmware and Bootloader.

BIOS Firmware

BIOS can either disable TPM2_Clear() command completely, or disable the interface of allowing software to “clear” all secrets created in Owner Hierarchy with physical presence request.

Such interface (to be disabled) could be a particular ACPI method implemented in UEFI BIOS in SMM mode. (Typically, OS or any software can issue this special ACPI method (via SMI), then BIOS SMM handler will set a non-volatile flag to indicate a physical presence request. After system restarts, the BIOS will promote a window and request user to confirm the TPM “clear” operation before proceeding to system boot.)

However, these modifications might be optional - i.e, if the product or project can allow users to clear all data for factory reset with physical presence, then BIOS doesn’t need such modifications.

Bootloader

Bootloader is required to do these below:

1. One-time tSeed Provisioning right after UEFI BIOS secure boot is enabled.
   Some fastboot commands are designed to provision Trusty tSeed in TPM NV Index.

   Policies (as previously mentioned) then will be configured by bootloader. Since PCR[7] measurement is used for access control, this must be done after UEFI BIOS secure boot is enabled. [*Note that how to enable / configure UEFI BIOS secure boot is out of scope in this document*]

   Bootloader changes the well-known Owner Hierarchy AuthValue (zeros) to a random

high-entropy number and discard it, in order to permanently lock / disable the Owner Hierarchy due to the fact that there is no TPM usage with Owner Hierarchy in Android AOSP implementation.

2. Run-time tSeed Handoff
   When access policy is satisfied, Bootloader will read the tSeed from TPM, and lock the Seed from further access (i.e. TPM2_ReadLock() and extend PCR[7]). Bootloader will then send tSeed to Hypervisor / Trusty after verification. The tSeed secret must also be wiped from memory of Bootloader after handoff.

5.4. Provisioning

As previously mentioned, Bootloader (fastboot in Android) is also responsible for provisioning the Trusty Seed into TPM NV index with proper protection. This should be done after UEFI Secure Boot is enabled and properly configured, and before shipping device to market.

In order to provision this Trusty Seed, two fastboot command options are introduced:

1. **Provisioning Trusty Seed**
   fastboot oem fuse provision-trusty-seed

   This command attempts to create (or delete and then re-create if the same NV index of Trusty Seed already exists) a TPM NV Index with appropriate attributes as shown above for storing the 256-bit Trusty Seed which is generated randomly.

   After that, this command also locks write access to this TPM NV Index permanently so that any future write access to it will be denied (unless the TPM NV Index is deleted for some reason).

2. **Locking TPM Owner Hierarchy**
   fastboot oem fuse lock-tpm2-owner

   The command above is used to provision the Trusty Seed NV Index into TPM, then this command is required to lock it in order to prevent future deletion or removal of such NV Index from TPM. This command will simply change the AuthValue of TPM Owner Hierarchy with a random value, then discard it. This is what we called as “lock” operations in this command.

   Note it should be run manually by user after Trusty Seed provisioning. Keep in mind there is no corresponding “unlock” command to reverse it back to unlocked state. Before performing the “lock” operations, this command will internally check whether UEFI Secure Boot is enabled, whether the Trusty Seed is created, and whether the attributes of the Trusty Seed NV index are configured correctly. Till all of the above conditional checks are true, the command will lock it.

6. TPM Rooted RPMB Seed

The first part of this document above provides the details of how to create TPM NV Index for Trusty Seed and how to protect it from illegitimate access request. This chapter focuses on RPMB Seed creation and protection.

RPMB Key (rKey) is a 256-bit long high-entropy secret, unique per storage device and even per RPMB partition in the same storage device. Any access to this rKey (or rSeed which derives rKey) is restricted by TPM policy control.

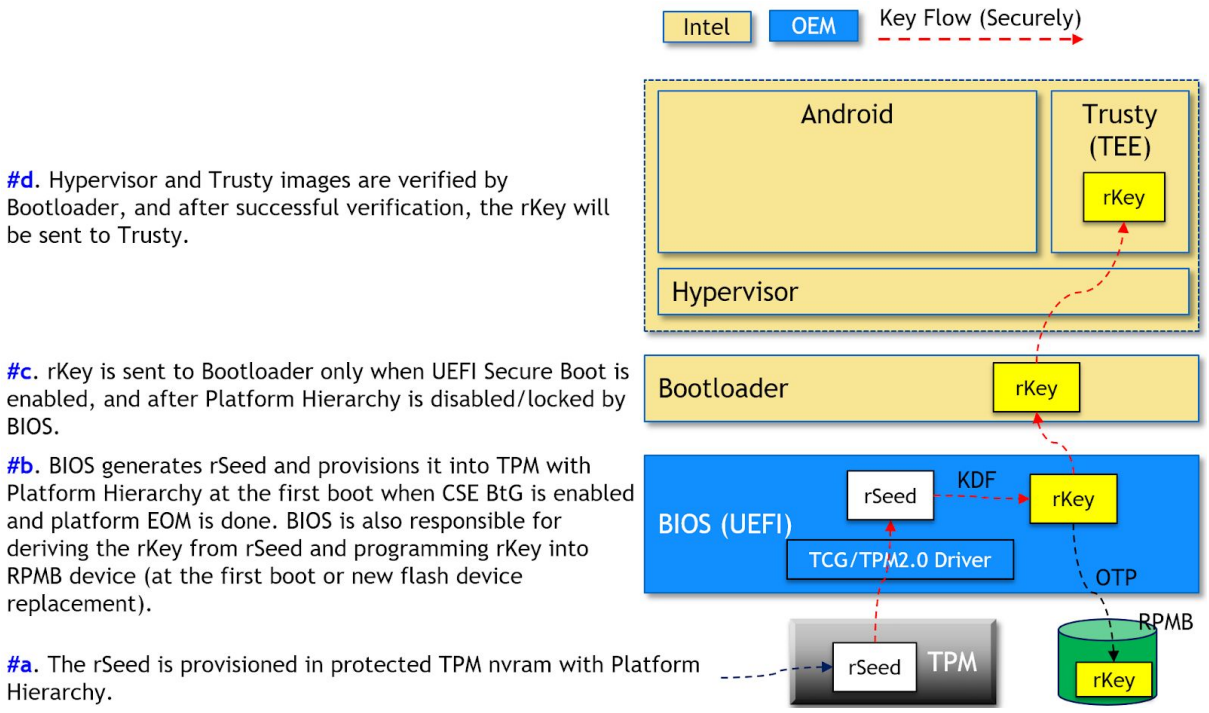
6.1. TPM Secure NV index with Platform Hierarchy

Unlike Trusty Seed which is used to protect user-generated data by Trusty and hence is managed under TPM Owner Hierarchy, RPMB Seed can be treated as platform resource because eMMC/UFS/NVM storage device is sort of platform configuration, which means that the RPMB should be able to be used by whatever OS's or whatever users on that particular platform. Besides, the RPMB Key can not be changed as long as it is programmed into the storage device. If it is cleared or lost, a new storage device should be used to replace the old one because the platform firmware / software will lose the write access to the RPMB partition forever.

Thus, TPM Platform Hierarchy is used to create and manage the TPM NV Index for RPMB Seed.

TPM-Rooted RPMB Seed NV Index

The TPM-rooted RPMB Seed / Key handoff flow is illustrated as the diagram below:



In this workflow, a rSeed is provisioned initially with a random 256-bit value into a TPM NV index with TPM Platform Hierarchy, and BIOS sets appropriate access control policy (see later) to restrict any illegitimate access to this secret. Also in the first boot, BIOS is responsible for deriving the rKey and programing it into RPMB device. In every boot, BIOS retrieves this secret when access policy is satisfied, and sends it to Bootloader when UEFI Secure Boot is enabled and Bootloader is verified with valid image signature.

TPM NV Index Attributes and Access Policy

TPM Platform Hierarchy can only be used by BIOS. In current Celadon reference platforms, it is limited to add code into BIOS firmware because it is owned by 3rd parties. In Celadon reference implementation, the RPMB key is derived from Trusty Seed (rSeed) which is created with TPM Owner Hierarchy instead.

However, in this section the recommendations are provided based on TPM Platform Hierarchy. So if there is any product / project that is using Celadon, the BIOS developers should design the corresponding implementation based on these recommendations below.

When creating a new NV Index with TPM2 command TPM2_NV_DefineSpace() with TPM Platform Hierarchy, the Attributes of this rSeed NV Index should be defined as shown in Table below.

Attribute Bit(s)	Value	Descriptions
TPMA_NV_PPREAD	1	The Index data can be read if Platform

		Authorization is provided.
TPMA_NV_PPWRITE	1	The Index data can be written if Platform Authorization is provided
TPMA_NV_WRITEALL	1	A partial write of the Index data is not allowed. The write size shall match the defined space size.
TPMA_NV_POLICY_DELETE	1	Index may not be deleted unless the authPolicy is satisfied using TPM2_NV_UndefineSpaceSpecial(). **Note** An Index with this attribute and a policy that cannot be satisfied (e.g., an Empty Policy) cannot be deleted. For rSeed NV Index, an Empty Policy is used to prevent it from being deleted.
TPMA_NV_WRITEDEFINE	1	TPM2_NV_WriteLock() may be used to prevent further writes to this location.
TPMA_NV_PLATFORMCREATE	1	This Index may be undefined with Platform Authorization but not with Owner Authorization. The TPM will validate that this attribute is SET when the Index is defined using Platform Authorization and will validate that this attribute is CLEAR when the Index is defined using Owner Authorization.
TPMA_NV_READ_STCLEAR	1	TPM2_NV_ReadLock() may be used to SET TPMA_NV_READLOCKED for this Index.

Note that all other attribute bit field value(s) should be clear/0.

With TPM NV Index attributes configured as above, the corresponding access control policy can be applied

Access	Policy Description
Read	After each TPM power cycle, read is allowed once and then the NV index is locked (until next power cycle) with TPM2_NV_ReadLock(). Besides, after Platform Hierarchy is disabled / locked by BIOS, any read access to this rSeed NV index will be denied.
Write	Write once right after rSeed provisioning, and then any write access is not allowed permanently, regardless of TPM power cycles.

	The TPM command is TPM2_NV_WriteLock().
Deletion	Deletion is not allowed by setting a TPMA_NV_POLICY_DELETE attribute bit with an EMPTY policy (zero digest) that can NEVER be satisfied for this NV Index. **Note** TPM2_Clear() won't be able to remove any NV Index created with Platform Hierarchy.

6.2. Security Guidelines

This section is intended to depict the security guidelines and considerations including security assets, security dependencies, the threats and mitigations.

Security Asset(s)

The asset here is RPMB Seed (rSeed) stored in TPM NV Index with Platform Hierarchy, and all the derivative keys from this rSeed, for example, the rKey(s).

The basic requirements for this rSeed asset are:

- It must be only accessible to the secure trusted entities. It should never be exposed to untrusted components. E.g. when UEFI Secure Boot is disabled,
- The rSeed temporarily buffered in the middle of the handoff shall be erased from memory after use as soon as possible, and it must not be stored in any ordinary storage on flash drive or disk.

Security Assumptions/Dependencies

Again, in Celadon project, since BIOS firmware is not fully controlled by Intel and it is developed by 3rd party OEM vendors for KBL/APL NUC devices on the market, we typically may do nothing but only provide our security mitigation recommendations to request their integration.

Threats and Mitigations

The table below lists all the identified threats and the corresponding mitigations:

#	Threat Descriptions	Mitigation(s)
1	DoS attack to remove / delete the rSeed NV Index from TPM. Impact: RPMB flash storage device must be replaced.	rSeed Index is created with Platform Hierarchy. So that TPM2_Clear() won't be able to clear it; Platform Hierarchy is disabled / locked by

		BIOS, so that all subsequent software modules (e.g. bootloader, OS, software…) cannot access this rSeed NV Index. rSeed NV Index is created with EMPTY policy (zero digest) which is never satisfied for removal.
2	UEFI Secure Boot is disabled, then get the RPMB Key by any UEFI applications	BIOS refuses to send rSeed/rKey to Bootloader as long as UEFI Secure Boot is disabled.
3	Secrets in RPMB can be read (even without rKey)	This is expected behavior in RPMB specification. Hence, do not write any plaintext secret data in RPMB, instead, the user data must be encrypted (e.g. with tSeed and/or combination with user credentials) before writing data to RPMB.
4	UEFI Secure Boot keeps being enabled, but secure boot configuration is changed to boot the alternative OS/Applications (e.g. switching OS from Android -> Windows -> Android) In this case, the malicious software can record the old data (although its content cannot be understood), and replay the old data by the new malicious alternative OS, and then the attacker switches back to previous victim OS software for replay attack.	Whenever UEFI secure boot configuration data is changed, BIOS will erase all the previous data / content in RPMB. Since there is no “erase” command for RPMB, BIOS can overwrite the entire RPMB data area with all zeros. **Note** BIOS firmware can get to know whenever the UEFI Secure Boot configurations or settings are changed.

6.3. Integration Guide

In order to integrate this feature (TPM-rooted RPMB Seed implementation), the modifications are required in BIOS firmware and Bootloader.

BIOS Firmware

BIOS should provision the rSeed at the first boot, when EOM (End of Manufacturing) is closed and CSE FW Boot Guard (BtG) is enabled, into TPM NV Index with TPM Platform Hierarchy, and apply the access restrictions as mentioned above.

BIOS will also derive the rKey from this rSeed with the device storage serial number as the derivation parameter. The recommended KDF (key derivation function) could be HMAC-SHA256 (https://tools.ietf.org/html/rfc4634#section-7, https://tools.ietf.org/html/rfc2104) or HKDF-SHA256 (https://tools.ietf.org/html/rfc5869).

Also, when BIOS detects RPMB key is not yet programmed on current storage drive, BIOS will program the rKey into RPMB device. Hence, RPMB driver is required in BIOS firmware implementation.

BIOS should only send the rKey to Bootloader (as per request) when UEFI Secure Boot is enabled. BIOS should implement a particular UEFI protocol interface to allow Bootloader to request rKey. However, the details of this protocol interface are not defined in this document.

Now that when a storage device is replaced (e.g. due to flash device malfunction), a new rKey should be generated and provisioned into the new RPMB device. Then the BIOS has two options to generate a different rKey:

1. Use the solution recommended previously
   rKey' = KDF (rSeed, new device serial number);

2. Delete the old rSeed NV Index, and re-create a new same NV Index with a different new rSeed value. Then use this rSeed as rKey (rKey = rSeed), in this option, there is no need key derivation in BIOS, and BIOS must NOT apply the "Deletion" policy as mentioned above to prohibit rSeed NV Index deletion.

Bootloader

In Celadon, Android Bootloader can use the rKey UEFI protocol interface mentioned above to retrieve the rKey for RPMB access.

7. References

1. TPM 2.0 library specification
   https://trustedcomputinggroup.org/resource/tpm-library-specification/
2. Unified Extensible Firmware Interface Specification version 2.6
   https://www.uefi.org/sites/default/files/resources/UEFI%20Spec%202_6.pdf